CLAIMS

1. A method for providing modular design in a programmable logic device, the method comprising:

partitioning a top-level logic design into a plurality of modules;

implementing each module using information generated by the partitioning step; and

assembling the modules using information generated from the implementing step and the partitioning step.

2. The method of Claim 1 wherein the step of partitioning includes positioning any global logic outside the plurality of modules.

3. The method of Claim 2 wherein the global logic includes at least one of the following resources: inputs/outputs, clock nets, a delay locked loop (DLL), and a random access memory (RAM).

4. The method of Claim 2 wherein the global logic includes resources that are not evenly distributed across the programmable logic device.

5. The method of Claim 1 wherein the step of partitioning includes sizing and positioning each module on the programmable logic device.

6. The method of Claim 5 wherein the step of partitioning includes positioning pseudo logic for each module, wherein pseudo logic for a first module is positioned outside a boundary defined for the first module.

7. The method of Claim 6 wherein the pseudo logic for the first module is positioned inside a boundary defined for an adjacent module.

8. The method of Claim 1 wherein the step of partitioning includes creating a physically implemented module (PIM) directory.

9. The method of Claim 1 wherein the step of partitioning includes publishing predetermined files to a centralized directory.

10. The method of Claim 9 wherein the predetermined files include module level native generic object, top level netlist constraints database, and top level native generic mapped files for the module.

11. The method of Claim 1 wherein the step of partitioning includes generating a first file comprising a description of the top-level logic design in primitive elements.

12. The method of Claim 11 wherein the top-level first file comprises a top-level EDIF file.

13. The method of Claim 11 wherein the step of partitioning includes generating a second file comprising inter-module timing constraints for the top-level logic design.

14. The method of Claim 13 wherein the second file comprises a top-level netlist constraints file (NCF).

15. The method of Claim 13 wherein the step of partitioning includes using the first and second files to generate a third file comprising the description of the top-level logic design, a hierarchy of the top-level logic design, and any constraints of the first and second files.

16. The method of Claim 15 wherein the third file comprises a top-level native generic object (NGO) file.

17. The method of Claim 15 wherein the step of partitioning includes using the third file to generate a fourth file comprising the description of the top-level logic design, the hierarchy of the top-level logic design, and any constraints of the top-level logic design.

18. The method of Claim 17 wherein the fourth file comprises a top-level native generic database (NGD) file.

19. The method of Claim 17 wherein the step of partitioning includes using information in the fourth file to generate a fifth file that includes inter-module constraints of the top-level logic design.

20. The method of Claim 19 wherein the fifth file further includes module-to-input/output constraints of the top-level logic design.

21. The method of Claim 19 wherein the fifth file comprises a top-level user constraints file (UCF).

22. The method of Claim 19 wherein the step of partitioning includes annotating the fourth file with information from the fifth file.

23. The method of Claim 1 wherein the implementation of at least two modules is performed substantially in parallel.

24. The method of Claim 1 wherein the implementation of the plurality of modules is performed in any order.

25. The method of Claim 1 wherein the step of implementing includes implementing each module in a separate module directory.

26. The method of Claim 1 wherein the top-level logic design provides context for the module implementation.

27. The method of Claim 1 wherein position and size of all modules are used in implementation of each module.

28. The method of Claim 1 wherein the top-level logic design includes positions of pseudo logic used for module implementation.

29. The method of Claim 1 wherein the step of implementing includes mapping each module.

30. The method of Claim 29 wherein mapping includes adding pseudo logic to any unconnected ports of at least one module.

31. The method of Claim 30 wherein the pseudo logic is not implemented in the top-level logic design.

32. The method of Claim 1 wherein the step of implementing includes adding pseudo logic to any unconnected ports of the module.

33. The method of Claim 1 wherein the step of implementing includes placing and routing each module.

34. The method of Claim 33 wherein the step of placing includes placing any unconstrained pseudo logic, and then placing module logic in the module.

35. The method of Claim 1 wherein the step of implementing includes floorplanning at least one module.

36. The method of Claim 36 wherein floorplanning the module includes placing any pseudo logic, and then placing module logic.

37. The method of Claim 35 wherein floorplanning necessitates mapping, placing, and routing at least one module another time.

38. The method of Claim 1 wherein the step of implementing includes simulating each module.

39. The method of Claim 38 wherein simulating is performed using the top-level logic design as context.

40. The method of Claim 38 wherein simulating is performed independently from the top-level logic design.

41. The method of Claim 38 wherein simulating the module includes simulating dangling signals of the module.

42. The method of Claim 1 wherein the step of implementing includes publishing predetermined files for each module to a centralized directory.

43. The method of Claim 42 wherein the centralized directory is a physically implemented module (PIM) directory.

44. The method of Claim 19 wherein the step of implementing includes using the fifth file to generate a sixth file comprising user constraints associated with a module and the top-level design.

45. The method of Claim 44 wherein the sixth file comprises a module-relative top-level user constraints file (UCF).

46. The method of Claim 44 wherein the step of implementing includes generating a seventh file comprising a description of the module in primitive elements.

47. The method of Claim 46 wherein the seventh file comprises a module-level EDIF file.

48. The method of Claim 46 wherein the step of implementing includes generating an eighth file comprising constraints for the module.

49. The method of Claim 48 wherein the eighth file comprises a module-level netlist constraints file (NCF).

50. The method of Claim 48 wherein the step of implementing includes using the seventh and eighth files to generate a ninth file comprising the description of the module, a hierarchy of the module, but no constraints of the module.

51. The method of Claim 50 wherein the ninth file comprises a module-level native generic object (NGO) file.

52. The method of Claim 50 wherein the step of implementing includes using the third, sixth, and ninth files to generate a tenth file comprising the description of the module and the top-level design, the hierarchy of the module and the top-level design, and any constraints of the module and the top-level design.

53. The method of Claim 52 wherein the tenth file comprises a module-relative top-level native generic database (NGD) file.

54. The method of Claim 52 wherein the step of implementing includes using information in the tenth file to add intra-module timing constraints to the sixth file.

55. The method of Claim 52 wherein the step of implementing includes annotating the tenth file with information from the sixth file.

56. The method of Claim 52 wherein the step of implementing includes mapping the tenth file.

57. The method of Claim 52 wherein the step of implementing includes generating an eleventh file comprising a physical design database of the module in the context of the top-level design.

58. The method of Claim 57 wherein the eleventh file comprises a module-relative top-level netlist circuit description (NCD) file.

59. The method of Claim 57 wherein the step of implementing includes generating a twelfth file comprising the description and mapping of the module in the context of the top-level design, the hierarchy of the module and the top-level design, and any constraints of the module and the top-level design.

60. The method of Claim 59 wherein the twelfth file comprises a native generic map file.

61. The method of Claim 57 wherein the step of implementing includes placing and routing the eleventh file.

62. The method of Claim 1 wherein the step of assembling includes mapping the top-level design using guide files generated during the step of implementing.

63. The method of Claim 1 wherein the step of assembling including placing and routing the top-level design using guide files.

64. The method of Claim 1 wherein the step of assembling includes using

a module native generic object file for any modules in a public directory of module physical implementations,

a top-level native generic object file, and

a top level user constraints file

to assemble

a top-level native generic database file comprising the description of any modules in the top-level design, the hierarchy of any modules in the top-level design, and any constraints of any modules in the top-level design.

65. The method of Claim 1 wherein the step of assembling includes using a plurality of top-level netlist circuit description files and one top-level native generic database file to assemble a physical design database of any modules in the top-level design.

66. The method of Claim 65 wherein the physical design database comprises a new, top-level netlist circuit description (NCD) file.

67. The method of Claim 65 wherein the step of assembling includes using a plurality of top-level netlist

circuit description files and one top-level native generic database files to generate a top-level native generic mapped file comprising the description and mapping of any modules and the top-level design, the hierarchy of any modules and the top-level design, and any constraints of any modules and the top-level design.

68. The method of Claim 67 wherein the step of assembling includes using the plurality of top-level netlist circuit descriptions to fully implement the top-level design.

69. A method for providing modular design for a logic design including multiple modules, the method comprising:

generating a set of top-level files for the logic design in a top-level directory;

copying a portion of the first set of top-level files to a plurality of module directories, each module directory for implementing a single module;

generating a set of module-relative top-level files for each module using the portion of the set of top-level files;

mapping, placing, and routing each module in its respective module directory;

publishing a portion of the set of module-relative top-level files from each module directory to an implementation directory; and

implementing the logic design in the top-level directory by accessing the set of module-relative top-level files in the implementation directory.

70. A method comprising:

creating a top-level design including unelaborated modules;

generating at least one top-level file for the top-level design;

using the at least one top-level file to implement each unelaborated module and to generate a module-relative top-level file for each implemented module; and

assembling the implemented modules using the module-relative top-level file of each module.

71. A method of implementing a logic design in a programmable logic device, the method comprising:

dividing the logic design into a plurality of modules;

building each module based on information generated during the step of dividing;

mapping each module based on information generated during the step of building;

placing and routing each module;

assembling predetermined files associated with any placed and routed modules; and

mapping, placing, and routing the logic design using the assembled predetermined files.

72. A graphical tool comprising:

a structure for drawing rectangles;

a structure for drawing lines;

an interpreter for recognizing that a line crossing a boundary of a rectangle represents an assignment of pseudo logic; and

a structure for associating a piece of pseudo logic located at the end of the line outside the rectangle with a module represented by the rectangle.

73. A graphical tool comprising:

a line;

a first end of the line connected to a predetermined point in a module; and

a second end of the line connected to a port of the module, the port located outside the module.

74. The tool of Claim 73 provided in a design flow of a programmable logic device.

75. The tool of Claim 74 wherein the module includes an unelaborated logic block.

76. A programmable logic device including logic implemented by configuration data, the configuration data being generated by the following steps:

providing a top-level design including a plurality of unelaborated modules;

generating a top-level file for the top-level design;

using the top-level file to build each unelaborated module and to generate a module-relative top-level file for each built module;

using the module-relative top-level file of each module to assemble the built modules; and

generating an output file including the assembled, built modules, wherein the output file provides the configuration data for the programmable logic device.

77. A method for providing modular design in a programmable logic device, the method comprising:

partitioning a top-level logic design having a plurality of paths into a plurality of modules, each of the modules having a plurality of ports for connecting to other modules; and

implementing the modules such that all the ports are registered, whereby critical paths in the design are all inside a module.

78. In a logic design to be implemented in a programmable logic device, a method of positioning modules of the design comprising:

representing elements and connections of the design on a computer monitor;

drawing a boundary around a portion of the design to enclose elements of the design within the boundary, thereby forming a module;

drawing a second boundary around a second portion of the design to enclose a second group of elements of the design, thereby forming a second module; and

repeating until all elements of the design are enclosed within a module.

79. The method of positioning modules of Claim 78 wherein the boundaries are rectangular.